Diffusion Transformers with Residual Adaptive Layer Normalization

Ge Wu[1,2*] **Minxing Luo**[2] **Yikai Ge**[1,2] **Lei Wang**[2] **Dandan Zheng**[3]
Rui Liu[3] **Libin Wang**[3] **Yuliang Zhan**[4] **Taihang Hu**[2] **Jingdong Chen**[3] **Xiang Li**[1,2†]
[1]NKIARI, Shenzhen Futian, [2]VCIP, CS, Nankai University,
[3]Ant Group, [4]Renmin University of China



Figure 1: **Excellent image quality through Residual Adaptive Layer Normalization.**

Abstract

Residual connections with PreNorm are the default design in modern transformers due to their strong optimization stability, but recent research has shown that they weaken the functional differentiation of deeper blocks and increase representational redundancy in later layers. This issue also appears in diffusion models using similar architectures, where the deep block is expected to support progressive denoising refinement rather than repeated computation over similar states. Motivated by this, we revisit cross-layer shortcuts in diffusion transformers and argue that their role should go beyond merely transferring shallow information, instead guiding deeper blocks to improve depth utilization. To this end, we propose ***Res****idual Adaptive Layer* ***Norm****alization* (**ResNorm**), which routes residual cross-layer shortcuts into the existing adaLN modulation. ResNorm preserves the optimization advantages of PreNorm residual connections, while reusing the native adaLN interface to transmit early-layer representations as modulation signals. This design naturally combines cross-layer history with the model's native condition, such as timestep and class embeddings. ResNorm allows the earlier structural representation to adaptively scale, shift, and gate deeper blocks. This not only produces more differentiated deep representations and reduces cross-layer similarity in later blocks, but also accelerates training convergence and improves performance. On ImageNet 256×256, SiT-XL/2 + ResNorm achieves up to **2**× and **36**× faster training than SiT-XL/2 + REG and SiT-XL/2 + REPA. Moreover, SiT-L/2 + ResNorm trained for only 400K iterations already outperforms SiT-XL/2 + REG. Code is available at: `https://github.com/Martinser/REG`.

*Work is done when Ge Wu is an intern at Ant Group.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

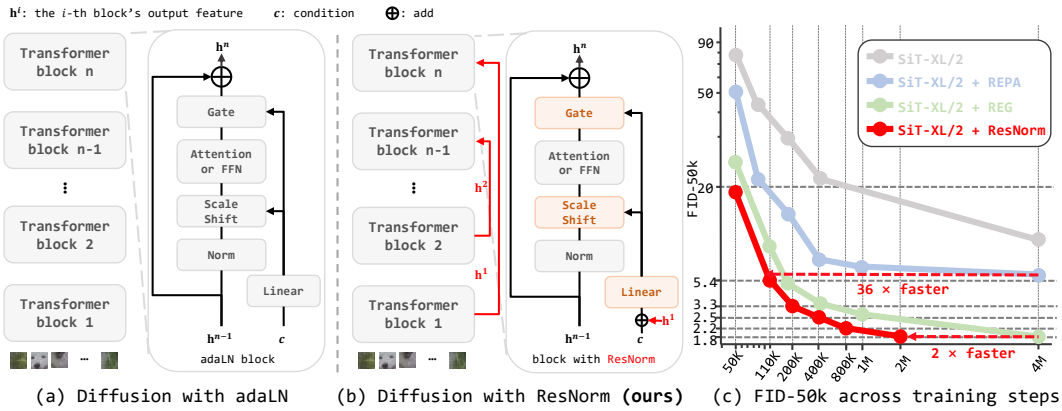


Figure 2: **Comparison between Adaptive Layer Normalization (adaLN) and Residual Adaptive Layer Normalization (ResNorm).** (a) In adaLN, each block is only modulated solely by the external condition $\mathbf{c}$. (b) ResNorm reuses the adaLN pathway to inject shallow features $\mathbf{h}^i$ as the residual cross-layer shortcut, enabling earlier representations to adaptively modulate deeper blocks. (c) ResNorm encourages more complementary refinement across depth, leading to faster training convergence and improved generation quality.

1 Introduction

Diffusion models have progressed rapidly [1, 2, 3, 4, 5], from early formulations such as DDPM [6] and DDIM [7] to transformer-based generators like DiT [8] and SiT [9], substantially improving both scalability and synthesis quality. More recently, a parallel line of work has shown that representation learning can further strengthen diffusion training. REPA [10] aligns diffusion features with foundation representations, REPA-E [11] trains tokenizer and diffusion end-to-end, and REG [12] jointly models low-level latents and high-level semantics. At the same time, efforts on unified tokenizers [13, 14, 15, 16] and pixel-space generation [17, 18, 19, 20] continue to broaden the design space of modern generative models.

Despite these advances in architecture, supervision, and tokenization, most diffusion transformers still use the same backbone recipe: standard residual connections [21] with PreNorm [22], which has become a standard for modern transformers [23, 24, 25], because it enables stable optimization at scale. However, this design introduces a subtle depth bottleneck: intermediate states are accumulated with unit weight, biasing the model toward shallow features and attenuating the effective contribution of deeper layers [26, 27]. Prior works address this either by redesigning residual pathways, such as residual scaling [28], hybrid normalization [26], and multi-stream connections [29, 30]. Another line of work tackles this by enhancing cross-layer communication through dense feature reuse [31], learned depth aggregation [32], and selective layer retrieval [27]. However, these solutions are largely tailored to language models. The same bottleneck persists in diffusion transformers, which use similar architectures (Figure. 2(a)). This motivates us to seek a minimal yet principled modification that preserves PreNorm's optimization benefits while substantially improving effective depth utilization in diffusion transformers.

In this paper, we propose ***Res**idual Adaptive Layer **Norm**alization* (**ResNorm**), which reuses the native adaLN pathway to route residual cross-layer shortcut as modulation signals (Figure. 2(b)). Motivated by the method of cross-layer communication [33, 34, 35], we revisit the design of cross-layer shortcuts in diffusion transformers (Section. 4.3) and argue that their role should not be merely to preserve shallow information through another additive path, but to improve effective depth utilization by guiding deeper denoising refinement. Accordingly, rather than injecting shallow representations through an additional residual branch, ResNorm allows earlier structural features to adaptively scale, shift, and gate deeper blocks, thereby preserving the optimization advantages of PreNorm while encouraging more differentiated computation across depth. Our analysis shows late-layer redundancy under standard PreNorm residual connections, whereas ResNorm consistently reduces cross-layer similarity and improves depth-wise specialization, resulting in notably superior image generation quality (Figure. 1). Empirically, on ImageNet 256×256, ResNorm achieves up to 2× and 36× faster

training than SiT-XL/2 + REG and SiT-XL/2 + REPA (Figure. 2(c)), and SiT-L/2 + ResNorm trained for only 400K iterations already surpasses SiT-XL/2 + REG (Table. 1).

In summary, our contributions are as follows:

- We revisit residual cross-layer shortcuts design in diffusion transformers from the perspective of depth-wise denoising refinement.
- We propose ResNorm, a simple design that routes cross-layer history into the adaLN modulation interface, enabling shallow representations to guide deeper denoising adaptively.
- We show that ResNorm reduces late-layer similarity, accelerates training convergence, and improves generative performance.

2 Related work

Generative models for image generation. Diffusion-based generative modeling has become the dominant paradigm for image synthesis, evolving from early stochastic and score-based formulations [1, 2, 3] to practical models such as DDPM/DDIM [6, 7] and latent diffusion [36]. More recently, efficiency-oriented variants, including consistency models, Flow Matching, and Rectified Flow [37, 4, 5], have substantially reduced the cost of training and sampling. Orthogonal to these advances in generative objectives, the backbone has progressively shifted from U-Nets to transformers, as exemplified by DiT and SiT [8, 9], with continued gains in both latent- and pixel-space generation [17, 18, 19, 20]. Within this architectural trend, U-ViT [33], MDTv2 [34], and Hunyuan-DiT [35] show that cross-layer shortcuts can improve feature reuse and information propagation across depth. Inspired by this line of work, we revisit cross-layer communication in transformer generators and introduce ResNorm, which routes cross-layer information through the existing adaLN modulation pathway rather than explicit feature fusion. In this way, ResNorm preserves the optimization benefits of PreNorm residual connections while enabling early-layer representations to modulate deeper computation with minimal architectural overhead.

Generative models with representation learning. Diffusion models also exhibit strong representation learning ability: their intermediate features encode transferable semantics and benefit tasks such as segmentation [38, 39, 40], depth estimation [41], and editing [42, 43, 44]. Prior work mainly improves these representations in two ways. One line augments diffusion training with external semantic supervision, including feature alignment to pretrained encoders [10], end-to-end VAE tuning [11], and discriminative guidance within denoising [12]. The other line explores unified tokenizers or latent spaces for understanding and generation, using stronger visual encoders or high-dimensional representation autoencoders to improve latent quality and reconstruction [13, 14, 15, 16]. While effective, these approaches largely focus on representation sources, supervision, or latent parameterization, rather than the internal transport of representations across denoising depth. As a result, how early semantic cues can be reused by deeper layers in a lightweight, optimization-friendly manner remains underexplored.

Depth rebalancing under normalization. PreNorm residual architectures [21, 22] are widely adopted in modern transformers [23, 24, 25] for their favorable optimization behavior. At the same time, an emerging line of work shows that this design can under-utilize depth: repeated identity accumulation with fixed unit coefficients tends to preserve shallow signals, while progressively weakening the relative influence of deeper transformations [26, 27]. Existing efforts mainly follow two directions. One line modifies the backbone itself, including residual scaling in DeepNet [28], hybrid normalization in Mix-LN [26], and multi-stream residual formulations such as Hyper-Connections [29] and mHC [30]. The other line improves inter-layer interaction through explicit cross-layer mechanisms, such as dense feature reuse [31], learned depth aggregation [32], and selective layer retrieval [27]. However, these advances are largely developed for language models. In diffusion transformers, the same issue is arguably more critical, since denoising depends not only on stable training but also on progressively refined representations across depth. This leaves a clear opportunity for a minimal and principled design that preserves the optimization advantages of PreNorm while improving effective depth utilization in diffusion models.

3 Method

We propose ResNorm, a simple method that routes residual cross-layer shortcuts through the native adaLN modulation pathway, allowing early representations to modulate deeper blocks while retaining the optimization benefits of PreNorm residual connections (Figure. 2(b)). Section. 3.1 reviews REG [12], which serves as the basis of our approach, and Section. 3.2 details the design of ResNorm.

3.1 Preliminaries

Our method builds upon REG [12], a diffusion framework that jointly models image latents and high-level semantics by incorporating foundation-model representations into the denoising process. Given an image $\mathbf{x}_*$, REG encodes it into a latent code $\mathbf{z}_*$ via a VAE encoder [36], and extracts a global class token $\mathbf{cls}_*$ along with dense visual features $\mathbf{f}_*$ from a pretrained vision encoder [45]. Then, REG corrupts both modalities under a shared continuous-time schedule $t \in [0, 1]$:

$$\mathbf{z}_t = \alpha_t \mathbf{z}_* + \sigma_t \boldsymbol{\epsilon}_z, \qquad \mathbf{cls}_t = \alpha_t \mathbf{cls}_* + \sigma_t \boldsymbol{\epsilon}_{\text{cls}}. \tag{1}$$

where α_t and σ_t are time-dependent interpolation coefficients, and $\boldsymbol{\epsilon}_z, \boldsymbol{\epsilon}_{\text{cls}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are Gaussian noise samples for the latent and semantic branches, respectively. The noised latent $\mathbf{z}_t$ is patchified into tokens $\mathbf{z}'_t$, and $\mathbf{cls}_t$ is linearly projected into $\mathbf{cls}'_t$. The two are concatenated to form the input sequence of the block:

$$\mathbf{h}_t = [\mathbf{cls}'_t; \mathbf{z}'_t]. \tag{2}$$

The model then processes $\mathbf{h}_t$ through n blocks, producing a sequence of hidden states. Denoting by the latent and class-token branches, REG is trained with a joint velocity objective:

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_{\mathbf{x}_*, \boldsymbol{\epsilon}, t}\left[\left\| \mathbf{v}(\mathbf{z}_t, t) - \dot{\alpha}_t \mathbf{z}_* - \dot{\sigma}_t \boldsymbol{\epsilon}_z \right\|^2 + \beta \left\| \mathbf{v}(\mathbf{cls}_t, t) - \dot{\alpha}_t \mathbf{cls}_* - \dot{\sigma}_t \boldsymbol{\epsilon}_{\text{cls}} \right\|^2 \right], \tag{3}$$

$\mathcal{L}_{\text{pred}}$ is the joint velocity prediction objective, where a shared velocity field $\mathbf{v}(\cdot, t)$ is trained to simultaneously capture both the latent and semantic trajectories at diffusion time t. Here, $\dot{\alpha}_t$ and $\dot{\sigma}_t$ denote the time derivatives of the interpolation coefficients α_t and σ_t, respectively, and β controls the relative weight of the class-token prediction term. Building on REPA [10], REG additionally aligns an intermediate hidden token sequence $\mathbf{h}_t$ with the target token sequence $\mathbf{y}_* = [\mathbf{cls}_*; \mathbf{f}_*]$ extracted from a pretrained visual encoder via a token-wise cosine similarity loss:

$$\mathcal{L}_{\text{REPA}} = -\mathbb{E}_{\mathbf{x}_*, \boldsymbol{\epsilon}, t}\left[\frac{1}{M} \sum_{m=1}^{M} \text{sim}\Big(\mathbf{y}_*^{[m]}, h_\phi(\mathbf{h}_t^{[m]}) \Big) \right], \tag{4}$$

$\mathcal{L}_{\text{REPA}}$ is an auxiliary representation alignment objective that encourages the intermediate hidden token sequence $\mathbf{h}_t^{[m]}$ to align with the target token sequence $\mathbf{y}_* = [\mathbf{cls}_*; \mathbf{f}_*]$. Since the two feature spaces are heterogeneous, each token in $\mathbf{h}_t$ is first projected by a learnable MLP head h_ϕ before token-wise cosine similarity is computed. Here, $(\cdot)^{[m]}$ indexes the m-th aligned token pair, M denotes the total number of aligned tokens, and λ controls the relative weight of the alignment loss in the overall training objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{REPA}}. \tag{5}$$

3.2 Residual adaptive layer normalization for generation

The key idea of ResNorm is to turn cross-layer shortcuts from additive feature injection into adaptive modulation. While PreNorm ensures stable optimization, its residual accumulation can reduce effective depth utilization and make later blocks increasingly redundant. Motivated by prior cross-layer communication designs [33, 34, 35], we revisit cross-layer shortcuts in diffusion transformers and reuse the native adaLN pathway to enable shallow representations to modulate deep blocks. Specifically, early structural features are used to adaptively scale, shift, and gate later layers, preserving the optimization benefits of PreNorm while promoting more differentiated computation across depth. Formally, given a diffusion transformer with n blocks, we select the first k shallow blocks as the source block and the last k deep blocks as the target block, establishing a reverse-order pairing between them:

$$\mathbf{B}^i \to \mathbf{B}^j, \qquad j = n - i + 1, \quad i \in \{1, \ldots, k\}, \tag{6}$$

where $\to$ denotes the representation $\mathbf{x}^i$ produced by the i-th shallow block $\mathbf{B}^i$ is routed to the receiving deep block $\mathbf{B}^j$. In other words, the first block informs the last block $\mathbf{B}^1 \to \mathbf{B}^n$, the second

block informs the penultimate block $\mathbf{B}^2 \to \mathbf{B}^{n-1}$, and so on. Accordingly, ResNorm is applied only to the deep blocks receiving it, while all earlier blocks remain unchanged.

Once the routing pattern is established, the next question is how the transmitted shallow history should be applied to each target block. We fuse the routed shallow representation $\mathbf{h}^i$ with the native diffusion condition $\mathbf{c}$, which is formed by adding the timestep embedding and the class embedding, to construct a history-aware modulation condition. This fused condition is then fed into the original adaLN modulation:

$$\boldsymbol{\gamma}^j_{\text{msa}}, \boldsymbol{\beta}^j_{\text{msa}}, \boldsymbol{\alpha}^j_{\text{msa}}, \boldsymbol{\gamma}^j_{\text{mlp}}, \boldsymbol{\beta}^j_{\text{mlp}}, \boldsymbol{\alpha}^j_{\text{mlp}} = f^j(\mathbf{c} + \mathbf{h}^i).chunk(6), \tag{7}$$

The output of f^j is evenly split into six chunks, corresponding to the scale, shift, and gate parameters for the attention and MLP branches, respectively. Here, $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$, and $\boldsymbol{\alpha}$ denote scale, shift, and gate. The f^j is the linear projection. Taking these history-aware modulation parameters into the standard diffusion transformer block yields the following update rule for each receiving deep block:

$$\mathbf{h}^j_t = \mathbf{h}^{j-1}_t + \boldsymbol{\alpha}^j_{\text{msa}} \odot \mathcal{F}^j_{\text{msa}}\big(\boldsymbol{\gamma}^j_{\text{msa}} \odot \text{LN}(\mathbf{h}^{j-1}_t) + \boldsymbol{\beta}^j_{\text{msa}}\big), \tag{8}$$

$$\mathbf{h}^j_t = \mathbf{h}^j_t + \boldsymbol{\alpha}^j_{\text{mlp}} \odot \mathcal{F}^j_{\text{mlp}}\big(\boldsymbol{\gamma}^j_{\text{mlp}} \odot \text{LN}(\mathbf{h}^j_t) + \boldsymbol{\beta}^j_{\text{mlp}}\big), \tag{9}$$

where $\mathcal{F}^j_{\text{msa}}$ and $\mathcal{F}^j_{\text{mlp}}$ denote the multi-head attention and MLP transformations. Crucially, unlike the standard diffusion block, where modulation parameters are derived solely from the embedding $\mathbf{c}_t$, here $\boldsymbol{\gamma}^j$, $\boldsymbol{\beta}^j$, and $\boldsymbol{\alpha}^j$ are further conditioned on the shallow features from the source block via ResNorm, enabling each deep block to adaptively incorporate cross-layer context through the existing modulation pathway without introducing additional sub-layers or modifying the other structure.

ResNorm enjoys three advantages: (1) **Optimization-preserving.** Because routed information enters only through the adaLN modulation rather than the other path, ResNorm retains the favorable gradient flow and training stability of PreNorm. Deep blocks thus remain easy to optimize, while becoming explicitly aware of the denoising history. (2) **History-guided specialization.** By exposing each late block to a distinct shallow-stage representation, ResNorm encourages a meaningful division of labor across depth: shallow layers capture diverse early evidence, whereas deep layers perform targeted refinement conditioned on that evidence. As a result, later blocks are less likely to collapse into similar functions, leading to stronger representational diversity and more effective denoising. (3) **Minimal yet effective.** ResNorm is a minimal architectural modification that reuses the existing adaLN pathway. It requires no extra parameters, yet still improves both convergence and quality.

4 Experiments

We research ResNorm from three complementary perspectives. **(1) Effectiveness.** Consistent performance gains across diverse tasks and settings are evaluated to validate the general applicability of ResNorm (Section. 4.2). **(2) Design.** Ablation studies on key components and comparisons against other architectural designs are conducted to justify each design choice (Section. 4.3). **(3) Mechanism.** The degree to which ResNorm alleviates depth-wise redundancy and promotes more diverse representations in deeper blocks is examined through in-depth analysis (Section. 4.4).

4.1 Setup

Training configuration. We follow training protocols of REG [12]. Experiments are conducted on the ImageNet dataset [57], with images preprocessed to 256×256 or 512×512 resolution via center cropping and resizing, following the ADM framework [46]. Each image is encoded into a latent representation using the Stable Diffusion VAE [36]. Model architectures B/2, L/2, and XL/2 (with 2×2 patch processing) follow the SiT specifications [9]. For comparability, we fix the training batch size to 256 and adopt identical learning rates and Exponential Moving Average (EMA) configurations as REG [12] and REPA [10]. Implementation details are provided in the Appendix.

Evaluation methodology. To comprehensively evaluate image generation quality across multiple dimensions, we employ a rigorous set of quantitative metrics including Fréchet Inception Distance (FID) [58] for assessing realism, structural FID (sFID) [59] for evaluating spatial coherence, Inception Score (IS) [60] for measuring class-conditional diversity, Precision (Pre.) for quantifying sample fidelity, and Recall (Rec.) [61] for evaluating coverage of the target distribution, all computed on a standardized set of 50K generated samples to ensure statistical reliability. Sampling follows REG [12] and REPA [10], using the SDE Euler–Maruyama solver with 250 steps.

Table 1: **Performance comparison across varying model sizes and training iterations.** All models are trained on ImageNet 256×256 without CFG.

Method	#Params	Iter.	FID↓
SiT-B/2	130M	400K	33.0
+ REPA	130M	400K	24.4
+ REG	132M	400K	15.2
+ ResNorm (ours)	132M	**400K**	**10.6**
SiT-L/2	458M	400K	18.8
+ REPA	458M	400K	9.7
+ REPA	458M	700K	8.4
+ REG	460M	100K	11.4
+ REG	460M	400K	4.6
+ ResNorm (ours)	460M	**100K**	**8.0**
+ ResNorm (ours)	460M	**200K**	**4.6**
+ ResNorm (ours)	460M	**400K**	**3.3**
SiT-XL/2	675M	400K	17.2
SiT-XL/2	675M	7M	8.3
+ REPA	675M	400K	7.9
+ REPA	675M	4M	5.9
+ REG	677M	400K	3.4
+ REG	677M	1M	2.7
+ REG	677M	2.4M	2.2
+ REG	677M	4M	1.8
+ ResNorm (ours)	677M	**200K**	**3.3**
+ ResNorm (ours)	677M	**400K**	**2.5**
+ ResNorm (ours)	677M	**800K**	**2.2**
+ ResNorm (ours)	677M	**2M**	**1.8**

Table 2: **Results comparison with SOTA methods on ImageNet 256×256.** All models are with CFG.

Method	Epochs	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
Pixel diffusion						
ADM-U [46]	400	3.94	6.14	186.7	0.82	0.52
VDM++ [47]	560	2.40	-	225.3	-	-
Simple diffusion [48]	800	2.77	-	211.8	-	-
CDM [49]	2160	4.88	-	158.7	-	-
PixelFlow-XL/4 [50]	320	1.98	-	282.1	0.81	0.60
DiP-XL/16 [18]	320	1.98	-	282.9	0.80	0.62
PixNerd-XL/16 [51]	320	1.95	-	298.0	0.80	0.60
PixelGen-XL/16 [52]	160	1.83	-	293.6	0.79	0.63
JiT-G/16 [20]	600	1.82	-	292.6	-	-
DeCo-XL/16 [17]	800	1.62	4.41	301.0	0.80	0.62
Latent diffusion, Transformer + U-Net hybrid						
U-ViT-H/2 [53]	240	2.29	5.68	263.9	0.82	0.57
DiffiT [54]	-	1.73	-	276.5	0.80	0.62
MDTv2-XL/2 [34]	1080	1.58	4.52	314.7	0.79	0.65
Latent diffusion, Transformer						
MaskDiT [55]	1600	2.28	5.67	276.6	0.80	0.61
SD-DiT [56]	480	3.23	-	-	-	-
DiT-XL/2 [8]	1400	2.27	4.60	278.2	**0.83**	0.57
SiT-XL/2 [9]	1400	2.06	4.50	270.3	0.82	0.59
+ REPA [10]	800	1.42	4.70	305.7	0.80	0.65
+ REG [12]	80	1.86	4.49	**321.4**	0.76	0.63
+ REG [12]	800	1.36	**4.25**	299.4	0.77	0.66
+ ResNorm (ours)	80	1.63	4.65	293.5	0.80	0.64
+ ResNorm (ours)	320	1.37	4.51	302.9	0.79	0.66
+ ResNorm (ours)	**400**	**1.34**	4.44	306.1	0.78	**0.66**

4.2 Improving the performance of generative models

Training convergence analysis. We first examine whether ResNorm improves training convergence. As shown in Table. 1, ResNorm consistently converges faster than both REPA and REG across model scales. On SiT-B/2, replacing REG with ResNorm reduces FID from 15.2 to 10.6 at the same 400K steps. More importantly, this advantage becomes increasingly pronounced as the model grows. On SiT-L/2, ResNorm achieves 4.6 FID at 200K iterations, matching REG at 400K with half the training cost. And it further improves to 3.3 at 400K, already surpassing SiT-XL/2 with either REPA or REG under the same iterations.

Table 3: **Computational cost and performance comparison.** ResNorm achieves the similar performance to REPA and REG using SiT-XL on ImageNet 256×256, while significantly reducing training steps.

Method	#Params	FLOPs↓	Training step↓	FID↓
REPA	675	114.46	4M	5.9
ResNorm	677	133.30	110K	5.4
REG	677	114.90	4M	1.8
ResNorm	677	133.30	2M	1.8

This trend continues on SiT-XL/2, where ResNorm achieves 2.2 FID at only 800K steps, matching the performance that REG attains only after 2.4M steps, corresponding to a 3× convergence speedup. Consequently, ResNorm improves both generation quality and training efficiency. As shown in Table 3, ResNorm matches the same performance of REPA and REG using only 2.75% and 50% of their respective training steps on 8×H20 GPUs, highlighting its remarkable training efficiency.

Table 4: **Comparison on ImageNet 512×512 with CFG.** ResNorm achieves the best performance among all methods.

Model	Epochs	FID↓
VDM++ [47]	-	2.65
ADM-G, ADM-U [46]	400	2.85
Simple diffusion (U-Net) [48]	800	4.28
Simple diffusion (U-ViT, L) [48]	800	4.53
MaskDiT [55]	800	2.50
DiT-XL/2 [8]	600	3.04
SiT-XL/2 [9]	600	2.62
SiT-XL/2 + REPA	200	2.08
SiT-XL/2 + REG	80	1.68
SiT-XL/2 + ResNorm (ours)	**80**	**1.58**

Comparison with SOTA models. Having established the optimization advantage of ResNorm in the unguided setting, we next examine whether this benefit persists under the more demanding classifier-free guidance (CFG), which uses the REG's same strategy [12]. As shown in Table. 2, ResNorm reaches the performance of REG trained for 800 epochs using only 320 epochs, and further improves at 400 epochs, yielding a clearly better quality–efficiency frontier. Overall, these results show that ResNorm is not merely a convergence aid but an improvement that remains highly effective in the practically guided generation regime.

Table 5: **Effect of source and target block.** Source blocks control the number of residual cross-layer shortcuts produced; target blocks determine where ResNorm is applied. All models use SiT-XL/2 trained for 50K iterations on ImageNet 256×256 without CFG. ↓/↑ denotes lower/higher is better.

Method	Source blocks	Target blocks	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
REPA	-	-	52.3	31.24	24.3	0.45	0.53
REG	-	-	26.7	16.49	59.2	0.60	0.54
ResNorm (Ours)	1∼6	23∼28	19.7	17.87	80.0	0.65	0.52
	1∼8	21∼28	19.2	17.69	81.2	0.66	0.53
	1∼10	19∼28	20.8	18.63	76.0	0.64	0.51
	1∼12	17∼28	21.1	18.68	75.8	0.64	0.52
	1∼14	15∼28	22.4	16.63	70.8	0.64	0.52
	1∼8	9∼16	21.2	11.52	74.5	0.64	0.55
	1∼8	12∼19	20.9	13.63	75.9	0.64	0.54
	1∼8	15∼22	20.4	14.39	44.7	0.64	0.54
	1∼8	18∼25	20.2	16.60	78.5	0.65	0.53
	1∼8	21∼28	19.2	17.69	81.2	0.66	0.53

Performance on other tasks. Beyond the main setting, we further evaluate ResNorm on ImageNet 512×512 with CFG. As shown in Table. 4, ResNorm surpasses both REPA and REG under the same 80-epoch training budget, showing its advantage in high-resolution generation. Overall, these results indicate that ResNorm is not a task-specific heuristic, but a general architectural improvement whose benefits transfer beyond the main benchmark.

4.3 Ablation Studies

Source block. We examine the optimal number of residual cross-layer shortcuts introduced by the source block, which equivalently determines how many deep blocks should be equipped with ResNorm. As shown in Table. 5, expanding the modulation range from 1∼6 to 1∼8 further improves performance, reducing FID from 19.7 to 19.2. This suggests that applying ResNorm to only a few of the deepest blocks is helpful, whereas covering the last eight blocks better aligns with the stage where representation homogenization may become most pronounced. However, extending ResNorm further to 1∼10, 1∼12, and 1∼14 degrades FID again, indicating that overly broad modulation starts to interfere with mid-depth representations that still need to preserve their own evolving semantics. Overall, these results reveal a sweet spot: ResNorm is most effective when it is applied to a sufficiently wide yet still localized late-stage region, with 1∼8 providing the best balance between performance and efficiency. Subsequent experiments will use 1∼8 as the default configuration of source blocks.

Target block. Building on the above analysis, we examine where ResNorm should be injected in the target block. As shown in Table. 5, with the source blocks fixed to 1∼8, shifting the target range progressively deeper consistently improves FID, from 21.2 at 9∼16 to 19.2 at 21∼28; the 21∼28 setting also achieves the best 81.2 IS and 0.66 Pre. This trend suggests that the benefit of ResNorm is most pronounced in the final denoising stage. A plausible reason is that early and middle blocks are still forming coarse semantics, so introducing shallow historical cues too early may interfere with their representation evolution. In contrast, deeper blocks are more prone to homogenization and are mainly responsible for detail refinement, where shallow features serve as more effective complementary corrections. Overall, these results confirm that ResNorm is most effective when applied to the deepest blocks. We adopt 21∼28 target blocks as the default configuration.

Table 6: **Ablation on routing strategy.** Each configuration corresponds to the routing variant in Figure 3. All models use SiT-B/2 trained for 400K iterations on ImageNet 256 × 256.

Method	Routing strategy	FID↓	sFID↓
REG	-	15.2	6.69
ResNorm (ours)	**(a) reverse 1-to-1**	**10.6**	**6.27**
	(b) straight 1-to-1	11.3	6.27
	(c) source-shared 1-to-N	10.9	6.28
	(d) target-shared N-to-1	13.7	6.83

Routing strategy. We examine the impact of different routing designs on the effectiveness of ResNorm. As shown in Figure. 3, we compare four topologies: **(a)** reverse 1-to-1, **(b)** straight 1-to-1, **(c)** source-shared 1-to-N, and **(d)** target-shared N-to-1. Table. 6 shows that (a) achieves the best FID of 10.6, outperforming (b), (c), and (d). This shows that the effectiveness of ResNorm is determined by the amount of reused shallow information and the routing topology itself. The superiority of (a) comes from two aspects. First, it preserves source diversity and target coverage: each deep block receives a distinct shallow signal, and the modulation is distributed throughout the late stage rather

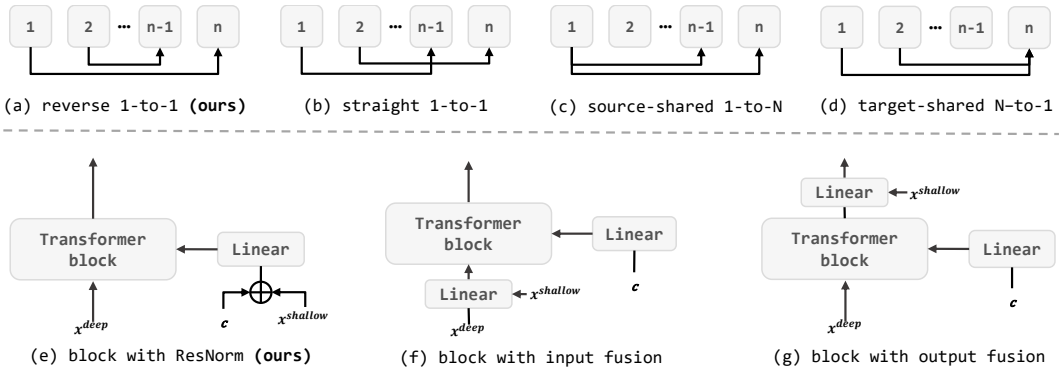


Figure 3: **Ablation study on the routing strategy and block-level fusion design of residual cross-layer shortcuts.** (a)–(d) We compare four routing strategies: (a) reverse 1-to-1 pairs shallow and deep blocks in reverse depth order; (b) straight 1-to-1 preserves the original depth order; (c) source-shared 1-to-N broadcasts one shallow feature to multiple deeper blocks; (d) target-shared N-to-1 aggregates multiple shallow features at a single target block. (e)–(g) We further compare three block designs, all built upon the routing strategy in (a), where x_{deep} and $x_{shallow}$ denote the input to the deep block and the output of the shallow block, respectively: (e) ResNorm performs cross-layer interaction through the existing modulation pathway, (f) input fusion injects routed features before the block, and (g) output fusion fuses them after the block.

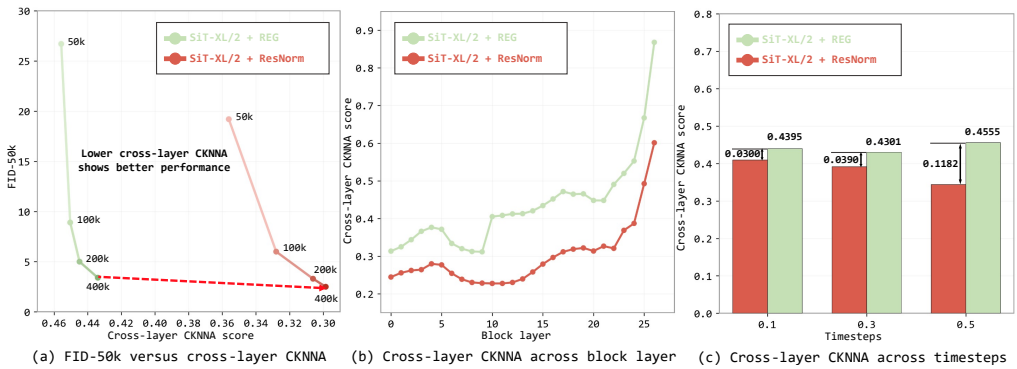
than duplicated or collapsed. Second, the reverse pairing is crucial. Since deeper blocks are more prone to representation homogenization, aligning them with earlier and more diverse shallow features provides the strongest complementarity. By comparison, (b) keeps straight 1-to-1 routing; (c) reuses the same shallow feature across multiple deep blocks, reducing source diversity; and (d) concentrates multiple shallow features on a target block, limiting the coverage of target blocks. Overall, these results suggest that ResNorm achieves the best performance when distinct shallow features are routed to distinct deep blocks in reverse order, maximizing both source diversity and target coverage.

Fusion design. To pinpoint how shallow information should be propagated across depth, we compare three designs in Figure. 3: (e) ResNorm, (f) input fusion, (g) output fusion. A clear pattern emerges: the two fusion strategies are both less effective than ResNorm, as they combine the shallow representation x^{shallow} with the deep representation x^{deep} either too early or too crudely. Input fusion injects x^{shallow} into x^{deep} before the transformer block performs its own refinement, forcing the block to reconcile heterogeneous semantics and thereby weakening depth specialization. Output fusion alleviates this interference by first refining x^{deep} and only then merging the resulting deep representation with x^{shallow}; however, it still treats the shallow pathway as a raw additive shortcut, leaving the scale and semantic mismatch between the two branches insufficiently resolved. ResNorm performs best for precisely this reason: rather than directly entangling x^{shallow} and x^{deep} at the input or output of the block, it uses x^{shallow} to modulate the conditioning pathway for transforming x^{deep}, preserving the autonomy of deep processing while ensuring that cross-level information remains complementary rather than competitive.

Table 7: **Ablation on transmission design.** Each design corresponds to Figure 3 (e)–(g). All models use SiT-B/2 trained for 400K iterations on ImageNet 256×256.

Method	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
REG	15.2	6.69	94.6	0.64	0.63
+ (e) ResNorm (ours)	**10.6**	**6.27**	**121.5**	**0.68**	**0.63**
+ (f) input fusion	13.0	6.23	104.5	0.68	0.62
+ (g) output fusion	12.9	6.16	105.4	0.68	0.63

ResNorm structure. We further ablate the internal design of ResNorm by enabling different combinations of the adaLN branches in Table. 8. Each branch alone already improves over the baseline, showing that shallow history is beneficial regardless of the injection form. Among them, the gate gives the largest single-branch gain. Yet the pairwise results show that scale and shift remain important complements: both improve performance when added to the gate. This indicates that the gate determines whether and how much shallow information should be injected, while scale and shift determine how the injected signal is aligned to the current feature statistics. Consequently, using all three yields the best overall result, achieving the lowest FID of 10.6 and the highest IS of 121.5. Overall, these results show that ResNorm works best with the full adaLN design.



(a) FID-50k versus cross-layer CKNNA (b) Cross-layer CKNNA across block layer (c) Cross-layer CKNNA across timesteps

Figure 4: **Cross-layer similarity analysis.** (a) FID versus cross-layer CKNNA, computed CKNNA between the final block feature and all preceding block features, averaged over both timesteps and layers. ResNorm consistently achieves lower CKNNA and better FID than REG in different train steps. (b) ResNorm yields lower CKNNA between the final block feature and each earlier block feature in 400K, averaged over the timesteps, indicating more differentiated late-layer representations across depth. (c) Averaged over depth, ResNorm at 400K remains consistently CKNNA below REG in different timesteps.

4.4 Improving the effective depth utilization

To understand why ResNorm performs better, we compute the CKNNA score [62] between the final block feature and preceding block features on ImageNet validation images at 256×256. We analyze this quantity across training steps, depth, and diffusion timesteps. Lower CKNNA indicates lower cross-layer feature redundancy. **Additional details are provided in the Appendix. A**

Table 8: Ablation design of different ResNorm structures. Each component contributes positively, and the full achieves the best. All results use SiT-B/2 trained for 400K on ImageNet 256×256.

Method	Gate	Scale	Shift	FID↓	sFID↓	IS↑
REG	✓	✓	✓	15.2	6.69	94.6
ResNorm (ours)	✓	-	-	11.6	6.44	115.4
	-	✓	-	12.7	6.51	109.1
	-	-	✓	12.5	6.41	108.3
	✓	✓	-	11.2	6.36	119.9
	✓	-	✓	11.0	6.27	119.4
	-	✓	✓	12.2	6.51	110.5
	✓	✓	✓	**10.6**	**6.27**	**121.5**

Across training steps. Figure. 4(a) reports the CKNNA score averaged over both timesteps and layers. We observe a clear correspondence between this quantity and FID: lower cross-layer CKNNA is usually associated with better generation quality. Notably, ResNorm remains below REG throughout training, indicating that it persistently alleviates late-layer redundancy and encourages deeper blocks to learn more distinct denoising functions, rather than repeatedly refining highly similar states.

Across depth. Figure. 4(b) visualizes the CKNNA between the final block and each intermediate block, averaged over the timesteps. For both models, similarity increases as layers approach the final block. However, ResNorm is consistently lower than REG across layers, suggesting that it systematically preserves functional differentiation in deep blocks.

Across timesteps. Figure. 4(c) further shows the CKNNA at different timesteps, averaged over the layers. ResNorm again yields uniformly lower scores than REG across timesteps, indicating that its advantage is robust to noise levels. This result further supports the ResNorm design: turning cross-layer shortcuts into modulation leads to more specialized computation.

5 Conclusion

This paper presents Residual Adaptive Layer Normalization (ResNorm). Instead of introducing another additive path to preserve shallow information, ResNorm reuses the native adaLN interface to transform early-layer representations into modulation signals that adaptively scale, shift, and gate deeper blocks. In this way, ResNorm preserves the optimization stability of PreNorm residual connections while reducing late-layer redundancy and encouraging more specialized denoising refinement across depth. Extensive experiments demonstrate that ResNorm consistently improves training convergence and generative performance.

Acknowledgements

This research was supported by Shenzhen Science and Technology Program (QNXMB20250701090801002, JCYJ20250604184027034), the Fund of the National Natural Science Foundation of China (Grant No. 62576177), Guangdong Basic and Applied Basic Research Foundation (2026A1515011435), the Fundamental Research Funds for the Central Universities 070-63263247, Youth Interdisciplinary Research Project of Nankai University (Grant No. NKQNJC202620) and the Tianjin Key Laboratory of Visual Computing and Intelligent Perception (VCIP). Computation is supported by the Supercomputing Center of Nankai University (NKSC).

References

[1] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. 2, 3

[2] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 2, 3

[3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2, 3

[4] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 2, 3

[5] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022. 2, 3

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 3

[7] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *arXiv preprint arXiv:2010.02502*, 2020. 2, 3

[8] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 2, 3, 6

[9] Nanye Ma, Mark Goldstein, Michael S Albergo, et al. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024. 2, 3, 5, 6

[10] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, et al. Representation alignment for generation: Training diffusion transformers is easier than you think. In *arXiv preprint arXiv:2410.06940*, 2024. 2, 3, 4, 5, 6, 14, 15

[11] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, et al. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. In *arXiv preprint arXiv:2504.10483*, 2025. 2, 3

[12] Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, et al. Representation entanglement for generation: Training diffusion transformers is much easier than you think. *arXiv preprint arXiv:2507.01467*, 2025. 2, 3, 4, 5, 6

[13] Minglei Shi, Haolin Wang, Wenzhao Zheng, Ziyang Yuan, Xiaoshi Wu, Xintao Wang, Pengfei Wan, Jie Zhou, and Jiwen Lu. Latent diffusion model without variational autoencoder. *arXiv preprint arXiv:2510.15301*, 2025. 2, 3

[14] Hao Tang, Chenwei Xie, Xiaoyi Bao, Tingyu Weng, Pandeng Li, Yun Zheng, and Liwei Wang. Unilip: Adapting clip for unified multimodal understanding, generation and editing. *arXiv preprint arXiv:2507.23278*, 2025. 2, 3

[15] Zhengrong Yue, Haiyu Zhang, Xiangyu Zeng, Boyu Chen, Chenting Wang, Shaobin Zhuang, Lu Dong, Yi Wang, Limin Wang, and Yali Wang. Uniflow: A unified pixel flow tokenizer for visual understanding and generation. *arXiv preprint arXiv:2510.10575*, 2025. 2, 3

[16] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders, 2025. 2, 3

[17] Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. *arXiv preprint arXiv:2511.19365*, 2025. 2, 3, 6

[18] Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. *arXiv preprint arXiv:2511.18822*, 2025. 2, 3, 6

[19] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. *arXiv preprint arXiv:2511.20645*, 2025. 2, 3

[20] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025. 2, 3, 6

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3

[22] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020. 2, 3

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2, 3

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 2, 3

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[26] Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*, 2024. 2, 3

[27] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026. 2, 3

[28] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024. 2, 3

[29] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024. 2, 3

[30] Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025. 2, 3

[31] Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. *arXiv preprint arXiv:2502.12170*, 2025. 2, 3

[32] Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024. 2, 3

[33] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*, 2023. 2, 3, 4

[34] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, et al. Mdtv2: Masked diffusion transformer is a strong image synthesizer. In *arXiv preprint arXiv:2303.14389*, 2023. 2, 3, 4, 6

[35] Zhimin Li, Jianwei Zhang, Qin Lin, Jiangfeng Xiong, Yanxin Long, Xinchi Deng, Yingfang Zhang, Xingchao Liu, Minbin Huang, Zedong Xiao, Dayou Chen, Jiajun He, Jiahao Li, Wenyue Li, Chen Zhang, Rongwei Quan, Jianxiang Lu, Jiabin Huang, Xiaoyan Yuan, Xiaoxiao Zheng, Yixuan Li, Jihong Zhang, Chao Zhang, Meng Chen, Jie Liu, Zheng Fang, Weiyan Wang, Jinbao Xue, Yangyu Tao, Jianchen Zhu, Kai Liu, Sihuan Lin, Yifu Sun, Yun Li, Dongdong Wang, Mingtao Chen, Zhichao Hu, Xiao Xiao, Yan Chen, Yuhong Liu, Wei Liu, Di Wang, Yong Yang, Jie Jiang, and Qinglin Lu. Hunyuan-dit: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding, 2024. 2, 3, 4

[36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, et al. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3, 4, 5

[37] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023. 3

[38] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, et al. Label-efficient semantic segmentation with diffusion models. In *arXiv preprint arXiv:2112.03126*, 2021. 3

[39] Wenliang Zhao, Yongming Rao, Zuyan Liu, et al. Unleashing text-to-image diffusion models for visual perception. In *ICCV*, 2023. 3

[40] Changyao Tian, Chenxin Tao, Jifeng Dai, et al. Addp: Learning general representations for image recognition and generation with alternating denoising diffusion process. In *arXiv preprint arXiv:2306.05423*, 2023. 3

[41] Ming Gui, Johannes Schusterbauer, Ulrich Prestel, et al. Depthfm: Fast monocular depth estimation with flow matching. In *arXiv preprint arXiv:2403.13788*, 2024. 3

[42] Amir Hertz, Ron Mokady, Jay Tenenbaum, et al. Prompt-to-prompt image editing with cross attention control. In *arXiv preprint arXiv:2208.01626*, 2022. 3

[43] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, et al. Diffusion autoencoders: Toward a meaningful and decodable representation. In *CVPR*, 2022. 3

[44] Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. *Advances in neural information processing systems*, 35:22117–22130, 2022. 3

[45] Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. Dinov2: Learning robust visual features without supervision. In *arXiv preprint arXiv:2304.07193*, 2023. 4, 14

[46] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 5, 6

[47] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In *NeurIPS*. 6

[48] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *ICML*. 6

[49] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *arXiv preprint arXiv:2207.12598*. 6

[50] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025. 6

[51] Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. *arXiv preprint arXiv:2507.23268*, 2025. 6

[52] Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Pixel diffusion beats latent diffusion with perceptual loss. *arXiv preprint arXiv:2602.02493*, 2026. 6

[53] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*. 6

[54] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. Diffit: Diffusion vision transformers for image generation. In *ECCV*. 6

[55] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. In *arXiv preprint arXiv:2306.09305*. 6

[56] Rui Zhu, Yingwei Pan, Yehao Li, Ting Yao, Zhenglong Sun, Tao Mei, and Chang Wen Chen. Sd-dit: Unleashing the power of self-supervised discrimination in diffusion transformer. In *CVPR*. 6

[57] Jia Deng, Wei Dong, Richard Socher, et al. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[58] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 5

[59] Charlie Nash, Jacob Menick, Sander Dieleman, et al. Generating images with sparse representations. In *arXiv preprint arXiv:2103.03841*, 2021. 5

[60] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. Improved techniques for training gans. In *NeurIPS*, 2016. 5

[61] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, et al. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. 5

[62] Minyoung Huh, Brian Cheung, Tongzhou Wang, et al. The platonic representation hypothesis. In *arXiv preprint arXiv:2405.07987*, 2024. 9, 14, 15

[63] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *arXiv preprint arXiv:2404.07724*, 2024. 15

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14

[65] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 14

[66] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, 2019. 14

Diffusion Transformers with Residual Adaptive Layer Normalization Supplementary Materials

A Cross-layer similarity details.

To complement Section. 4.4, we clarify how the reported CKNNA scores are computed and aggregated. For each training checkpoint s, we evaluate the model on ImageNet validation images at 256×256 over a set of p diffusion timesteps

$$\mathcal{T} = \{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}, \qquad p = |\mathcal{T}|.$$

Let $\mathbf{h}_t^{(s,l)}$ denote the feature of block l at timestep t, where the network has n blocks and the final block is indexed by n. We then compute the cross-layer similarity between the final block feature and each preceding block feature as

$$c_t^{(s,l)} = \mathrm{CKNNA}\left(\mathbf{h}_t^{(s,n)},\ \mathbf{h}_t^{(s,l)}\right), \qquad l = 1, \ldots, n-1,\ \ t \in \mathcal{T}.$$

Each checkpoint therefore yields $p(n-1)$ CKNNA scores in total. Lower values indicate lower cross-layer feature redundancy. From this perspective, Figure. 4(a)–(c) correspond to three complementary aggregations of the same score tensor.

Training-step view. For Figure. 4(a), we average over both timesteps and layers:

$$\bar{c}^{(s)} = \frac{1}{p(n-1)} \sum_{l=1}^{n-1} \sum_{t \in \mathcal{T}} c_t^{(s,l)}.$$

This yields one scalar CKNNA value for each training checkpoint s, which is then compared with the corresponding FID.

Depth-wise view. For Figure. 4(b), we average over timesteps only:

$$\bar{c}^{(s,l)} = \frac{1}{p} \sum_{t \in \mathcal{T}} c_t^{(s,l)}, \qquad l = 1, \ldots, n-1.$$

This gives $n-1$ values, one for each intermediate block, and reveals how similarity to the final block evolves across depth.

Timestep-wise view. For Figure. 4(c), we average over layers only:

$$\bar{c}_t^{(s)} = \frac{1}{n-1} \sum_{l=1}^{n-1} c_t^{(s,l)}, \qquad t \in \{0.1,\ 0.3,\ 0.5\}.$$

This gives p values, one for each evaluated timestep, and characterizes how cross-layer similarity changes along the denoising trajectory.

B Additional results

Table. 9 presents a detailed CFG ablation study of SiT-XL/2 + ResNorm, further demonstrating the consistent advantage of ResNorm across different classifier-free guidance scales.

C Experimental setup

Hyperparameter setup. Table. 10 summarizes the hyperparameter settings for ResNorm across different model scales. Following the experimental protocol of REPA [10], we optimize all models with AdamW [64, 65], using a learning rate of 1×10^{-4}. We further adopt DINOv2-B [45] as the alignment model and keep the number of denoising steps fixed at 250 for all inference runs.

CKNNA score. Centered Kernel Nearest-Neighbor Alignment (CKNNA) [62] provides a more fine-grained measure of representational alignment between neural networks. Compared with standard Centered Kernel Alignment (CKA) [66], it emphasizes local neighborhood consistency by evaluating

Table 9: The results of SiT-XL/2 + ResNorm at 2M training iterations under varying classifier-free guidance scale w, employing the guidance interval method [63].

Model	#Params	Iter.	Interval	w	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
ResNorm	677M	2M	[0, 0.8]	2.3	1.37	4.51	288.8	0.78	0.66
ResNorm	677M	2M	[0, 0.85]	2.3	1.34	4.44	306.1	0.78	0.66
ResNorm	677M	2M	[0, 0.9]	2.3	1.54	4.40	337.3	0.80	0.64
ResNorm	677M	2M	[0, 0.85]	2.4	1.35	4.47	309.3	0.78	0.66
ResNorm	677M	2M	[0, 0.85]	2.3	1.34	4.44	306.1	0.78	0.66
ResNorm	677M	2M	[0, 0.85]	2.2	1.34	4.42	302.6	0.78	0.65
ResNorm	677M	2M	[0, 0.85]	2.1	1.35	4.41	298.3	0.78	0.66

the agreement of nearest-neighbor structures in the representation space. In what follows, we largely use the notation from the original formulation [62]. Formally, given two kernel matrices K and L computed from the representations of two models, CKNNA is defined in three steps. First, a truncation function

$$\alpha(i, j) = 1\left[\phi_j \in \mathrm{knn}\left(\phi_i\right) \wedge \psi_j \in \mathrm{knn}\left(\psi_i\right)\right]$$

is introduced to retain only those pairwise relations for which sample j belongs to the k-nearest neighbors of sample i in both representation spaces, yielding a sparse kernel that focuses on local structure. Next, the weighted covariance between the truncated centered kernels is computed as

$$\mathrm{Align}_{\mathrm{knn}}(K, L) = \sum_{i,j} \alpha(i, j) \cdot \left(\bar{K}_{ij} \cdot \bar{L}_{ij}\right), \tag{10}$$

where $\bar{K}_{ij}$ and $\bar{L}_{ij}$ are the centered kernel entries. Finally, the alignment score is normalized to remove scale effects:

$$\mathrm{CKNNA}(K, L) = \frac{\mathrm{Align}_{\mathrm{knn}}(K, L)}{\sqrt{\mathrm{Align}_{\mathrm{knn}}(K, K) \cdot \mathrm{Align}_{\mathrm{knn}}(L, L)}}. \tag{11}$$

Following the CKNNA computation protocol introduced in REPA [10], we measure the representational similarity between the final block and each of the preceding blocks to quantify cross-layer redundancy.

Table 10: Hyperparameter settings across model scales.

Backbone	SiT-B	SiT-L	SiT-XL
Architecture			
#Params	132M	460M	677M
Input	32×32×4	32×32×4	32×32×4
Layers	12	24	28
Hidden dim.	768	1,024	1,152
Num. heads	12	16	16
β	0.03	0.03	0.03
λ	0.5	0.5	0.5
Alignment depth	4	8	8
$\mathrm{sim}(\cdot,\cdot)$	cos. sim.	cos. sim.	cos. sim.
Encoder $\mathcal{E}_{VF}(I)$	DINOv2-B	DINOv2-B	DINOv2-B
ResNorm settings			
residual cross-layer shortcuts	6	6	8
Optimization			
Batch size	256	256	256
Optimizer	AdamW	AdamW	AdamW
lr	0.0001	0.0001	0.0001
(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Interpolants			
α_t	$1-t$	$1-t$	$1-t$
σ_t	t	t	t
w_t	σ_t	σ_t	σ_t
Training objective	v-prediction	v-prediction	v-prediction
Sampler	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama
Sampling steps	250	250	250

D Limitations

Due to computational constraints, extended training of ResNorm across more iterations and broader configurations is left for future work.

E Broader Impacts

ResNorm improves training efficiency and generation quality in diffusion models, with potential benefits for applications. We encourage responsible deployment alongside appropriate safeguards.

F More visualization results

Additional visualization results of ResNorm with CFG ($w = 4.0$) are provided in Figures 5–24.



Figure 5: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Great white shark" (2).



Figure 6: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Bald eagle" (22).



Figure 7: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Great grey owl" (24).



Figure 8: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Macaw" (88).



Figure 9: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Sulphur-crested cockatoo" (89).



Figure 10: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Koala" (105).



Figure 11: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "American coot" (137).



Figure 12: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Lesser panda" (156).



Figure 13: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Border collie" (232).



Figure 14: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Timber wolf" (269).



Figure 15: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Polecat" (358).



Figure 16: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Lesser panda" (387).



Figure 17: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Giant panda" (388).



Figure 18: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Castle" (483).



Figure 19: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "China cabinet" (495).



Figure 20: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Convertible" (511).



Figure 21: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Bubble" (971).



Figure 22: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Geyser" (974).



Figure 23: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Lakeside" (975).



Figure 24: The visualization results of SiT-XL/2 + ResNorm use CFG with $w = 4.0$, and the class label is "Volcano" (980).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the paper's scope and contributions, with a strong focus on our key findings and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include theoretical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See the supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See the supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See the supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: This line of experimentation represents a distinct research direction from ours.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See the experiment part and supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, our research methodology has been conducted in full accordance with the ethical guidelines established by NeurIPS.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the supplemental material.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Null.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The construction of derivative datasets and subsequent research applications were predicated on obtaining express written consent from all originating data stewards. Each contributing entity formally validated their approval via email correspondence, unequivocally supporting our academic pursuits. These verified permission records, with all personally identifiable information redacted, will be disclosed alongside the published work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have meticulously recorded the complete experimental methodology and manuscript development procedures.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Null.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Null.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Null.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.